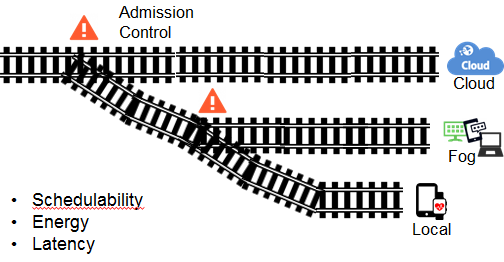
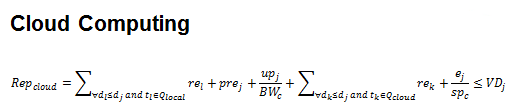
**論文內容**

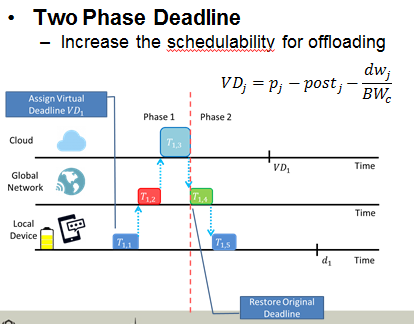
●系統架構:

●**OFLD** : 根據task特性去計算local執行會產生的energy以及cloud執行local需付出的energy來決定是不是要做offload，如果到cloud執行的resp time比local執行的exec還長，則offload到fog去做，來達到省能又有較好的resp time。必須配合admission control來確認是否能夠在該平台上執行，若否，則尋找其他平台或是回本地執行。 每當決定要到哪裡執行須須經過admission control的確認，包括本地端執行也是。

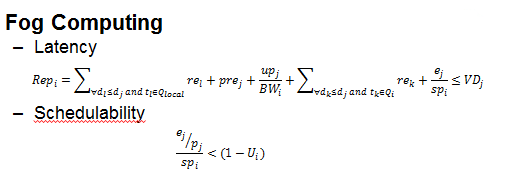


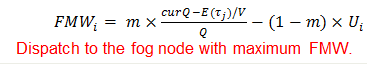


●**Virtual deadline:** 針對offloading task給予一個虛擬的截止時間，該截止時間的意義在於保留下載時間，也就是在虛擬截止時間之前必須完成遠端的運算才有時間做下載回本地端而不miss。



●**Fog Dispatcher** : 針對每個fog node，配合admission control確認是否能夠執行，並且會計算migration weight，選擇最大weight且utilization能夠容得下的node作為offloading目標，並且在決定目標時，會將task資訊給該目標做預訂，當同時間有其他人也要求同樣目標時，在admission會將此資訊算入，避免同時間有多個node將task offload到同一目標而造成miss。





●實驗設定:

* 在不同的task number下做不同的workload比較

(task number 🡪 3, 4 ,5 ,6, 8, 10)

(workload 🡪 0.5, 1.0, 1.5, 2.0)

* 固定task nunber及workload在不同bandwidth條件下的比較

(task number 🡪 5, workload🡪1.0)

(BW 🡪 1.0, 0.75, 0.5)

* 固定task nunber及workload在不同Migration weight (有兩組workload及battery的設定)

(task number 🡪 5, workload🡪1.0 : 1.0 : 0, batt🡪1.0 : 1.0 : 1.0) ,

(task number 🡪 5, workload🡪1.0 : 0.75: 0.25, batt🡪1.0 : 0.75 : 0.25)

(m🡪 0.2, 0.5, 0.8)

●數據分析:

* **Meet ratio:** 以load角度看，MEI🡪demand pkt，Greedy不考慮load(都是調到最小)。EIMA在高rate會接近Greedy。LDC low prio會被delay到後面，在高rate下，RR較LDC高。
* **Energy:** 看connection interval，連傳的次數。 LDC(各自節點週期除節點數)連傳績較Greedy(最小週期除節點數)少(LDC容易被block)，較省電。Polling有優先權概念，高prio次數相對較多，生命週期較RR差。

**模擬內容**

**實作**

●檔案區分

避免路徑引起的問題要放C:\

基本功能在basic func setting.docx，mesh在mesh.docx

目前是1個event1個pkt，若要增加pkt，則在創建event那邊丟幾次buf，gateway端抓最後一個pkt就可以得到最後的執行時間。

判斷旗標在第9 byte，0xFF代表是最後一個封包，這會在notification那邊判斷event id來設定結尾。

Gateway端則在HCI解event裡加入RSPDict[20:22]判斷是否為0xFF才繼續下面判定meet與否。

※注意在gateway端是big endian排列，在IAR是little endian

●硬體設定

Keyfob debugger有點接觸不良多試幾次

●量測方法

10歐姆電阻量測